REMARKS

In a Final Office Action dated July 26, 2005, the Examiner rejected claims 1, 3-11 and 13-24 under 35 U.S.C. 102(e) as anticipated by Wygodny (US 6,282,701 B1); and rejected claims 2, 12 and 25-27 under 35 U.S.C. 103(a) as unpatentable over *Wygodny* in view of Lindsey (US 5,896,536). Applicants filed a Notice of Appeal in response to the Final Office Action on November 28, 2005. The present amendment accompanies a Request for Continued Examination, and is a response to the Final Office Action of July 26, 2005.

Applicants have cancelled claims 13-24 and the rejections thereof are moot.

Applicants have amended the remaining independent claims (i.e., claims 1, 12 and 25) to further clarify the scope of the claimed invention. In particular, all independent claims as amended recite monitoring execution of the process or program to detect multiple occurrences of a reference or access to a memory location corresponding to the triggering expression, and collecting multiple respective instances of state data (sometimes called "snapshots") responsive to each triggering event to build a history of the triggering expression. Claims 3 and 6 have ben cancelled as superfluous, since their subject matter is now incorporated into the base claim. Various other dependent claims have been amended to conform to changes in the base claim. As amended, the claims are patentable over the cited art.

Various techniques and mechanisms for tracing the execution of computer programs are known in the art. Applicants' invention is intended to augment, rather than replace, the conventional set of tracing and debug tools. Applicants observed that, using conventional tools, it is often difficult to trace the history of a particular expression used in a program. An expression can be a declared variable, but is more often a pointer, array element, or instance of data structure or data within a data structure.

Specifically, the most common conventional trace technique involves identifying particular code statements, sometimes called break points or trace points, for collecting data. During execution of the program being traced, these code statements are monitored, and a defined set of trace data is collected whenever the corresponding code statement is encountered during execution. This is the basic technique disclosed in *Wygodny*. It is further possible to refine the technique by imposing conditions on the collection of data.

Unfortunately, if one simply wants to trace the history of a particular expression during program execution, the use of code statements as break points or trace points is less than optimal. It is theoretically possible to identify each and every statement in the code which might affect the particular expression, but this is often tedious and error prone. Furthermore, in the case of many types of expressions, the same code statement will be used for multiple different instances of a data type. For example, a single code statement might be used to access each element of an array. If the user is interested in the history of a single array element, collecting data from all occurrences of that single code statement will collect data any time any element of the array is accessed, potentially generating huge amounts of data. True, it is theoretically possible for the user to sift through all this data to find only those accesses to the array element of interest, but this is a lot of work. This problem is particularly acute when using object-oriented programming techniques.

Applicant's invention provides the program developer with an alternative tool, whereby it is possible for a developer to easily generate a narrowly defined set of data to characterize the history of a particular expression. In accordance with applicant's invention, generating a history of an expression is greatly simplified from the user's perspective by a tracing device which receives a specification of the expression ("trigger expression") from the user, and monitors ***accesses to the memory location representing the trigger expression*** during program execution. Appropriate state data is generated responsive to accessing the memory location of the user-

specified trigger expression, and a history of the trigger expression is thus constructed for later presentation to the user. Thus, an important feature of applicants' invention and distinction over the cited references is the nature of the triggering event.

Applicant's representative claim 1, as amended, recites:

1. A method of tracing the activity of an expression, said method comprising the steps of:
(a) specifying a machine-implemented process in which a trigger expression is to be traced;
(b) specifying the trigger expression to be traced in the machine-implemented process;
(c) monitoring execution of said machine-implemented process to *detect occurrences of a plurality of references to a location in machine memory representing a state of said trigger expression* occurring as a result of executing said machine-implemented process;
(d) *responsive to each detected occurrence of a reference to said location in machine memory* representing a state of said trigger expression, *storing the respective state of the trigger expression* at the time of the respective detected occurrence of a reference to said location in machine memory representing a state of said trigger expression *to create a history of said trigger expression within the machine-implemented process*, said storing step being performed without interrupting the machine-implemented process; and
(e) restoring the state of the trigger expression when requested. [emphasis added]

The remaining independent claims, while not identical in scope, contain analogous limitations to those italicized above.

Wygodny, the primary reference cited herein, discloses a tracing system for remote use, in which trace files can be generated during program execution for later analysis at a remote location. *Wygodny* loosely refers to "tracing" values of state variables. However, upon careful reading, it is clear that *Wygodny* is referring to the ***data that is collected with the trace, not the event which triggers collection of data***. Consistent with most conventional tracing tools, *Wygodny* discloses that a developer selects trace points ***in the code***, i.e. lines of code. These lines of code are the triggering events. During execution, trace data is collected upon encountering one

of the lines of code selected by the developer. The data that is collected is the state data specified by the developer, which may include local variables, global variables, static variables, etc.

Thus, *Wygodny* fails to disclose monitoring execution to ***detect references (or accesses) to a memory location representing the trigger expression***, and storing the state of the trigger expression responsive to detecting such a reference, as recited in applicants' amended independent claims. Accordingly, the claims as amended are not anticipated by *Wygodny*.

Nor are the amended claims obvious over *Wygodny*. The thrust of *Wygodny* is the collection of trace data in a remote or distributed environment. Although almost any state data might conceivably be collected, there is simply nothing in *Wygodny* that would suggest collecting state data of a triggering expression responsive to detecting a memory reference to the memory location storing the value of the triggering expression.

Lindsey, the secondary reference, discloses intercepting messages intended for specific data in an object-oriented program, and capturing trace data depending on the parameters of the message. *Lindsey* is cited to show imposing conditions on the triggering expression. I.e., according to *Lindsey*, conditions can be attached to the collection of trace data from an intercepted message. *Lindsey* does not teach or suggest, alone or in combination with *Wygodny*, that the triggering event which causes state data to be collected is a reference to a memory location corresponding to a triggering expression input by a user.

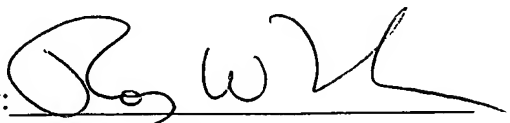
New independent claim 28 contains limitations analogous to those discussed above with respect to claim 1, and is patentable over the art of record for similar reasons.

In view of the foregoing, applicants submit that the claims are now in condition for allowance, and respectfully request reconsideration and allowance of all claims. In addition, the

Examiner is encouraged to contact applicants' attorney by telephone if there are outstanding issues left to be resolved to place this case in condition for allowance.

Respectfully submitted,

CARY L BATES, et al.

By: 

Roy W. Truelson
Registration No. 34,265

Telephone: (507) 202-8725